Techniques for Enabling Highly Efficient Message Passing on Many-Core Architectures

Min Si
University of Tokyo, JAPAN
msi@il.is.s.u-tokyo.ac.jp

Pavan Balaji (Co-advisor)
Argonne National Laboratory, USA
balaji@mcs.anl.gov

Yutaka Ishikawa (Advisor)
RIKEN AICS, JAPAN
yutaka.ishikawa@riken.jp

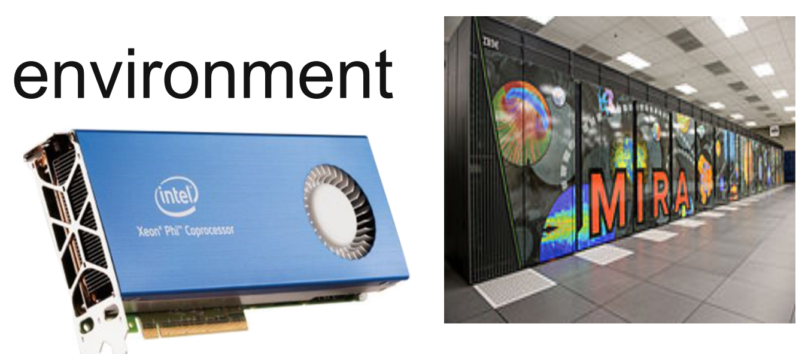
In this thesis, we investigate the characteristics of MPI on massively parallel many-core architectures and propose efficient strategies for two most popular programming models:

- **Improving core utilization and communication** for the hybrid MPI+thread model;
- **Efficient process-based asynchronous progress** for the MPI one-sided communication model.

Background

Many-Core

- Massively parallel environment
- Intel® Xeon Phi
- Blue Gene/Q



Hardware Characteristics

- Simple & low frequency core design
- #core increases faster than other on-chip resources

Parallelism and Resource Sharing in Scientific Applications

1. Hybrid MPI+threads model
✓ Massive parallelism in computation
✓ On-chip resource sharing between threads

2. MPI RMA-based PGAS model
✓ Memory sharing across nodes

Part I. Multithreaded MPI for Hybrid MPI+Threads Model

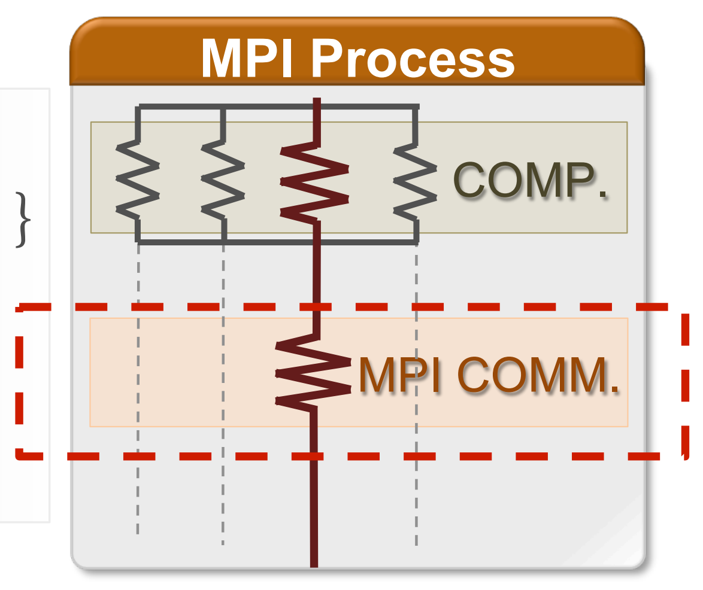
Problem in hybrid MPI + Threads

- Thread **Funneled** or **Serialized** mode
- Multiple threads parallelize computation
- Only **one thread issues MPI calls**
- **Most cores are IDLE during MPI Calls**

Funneled Mode

```
#pragma omp parallel
{ /* User Computation */ }

MPI_Function ( );
```



Solution: MT-MPI — Sharing Idle Threads with Application inside MPI [1]

```
#pragma omp parallel
{ /* User Computation */ }

MPI_Function( ){
  #pragma omp parallel
  { /* Internal Processing */ }
}
```

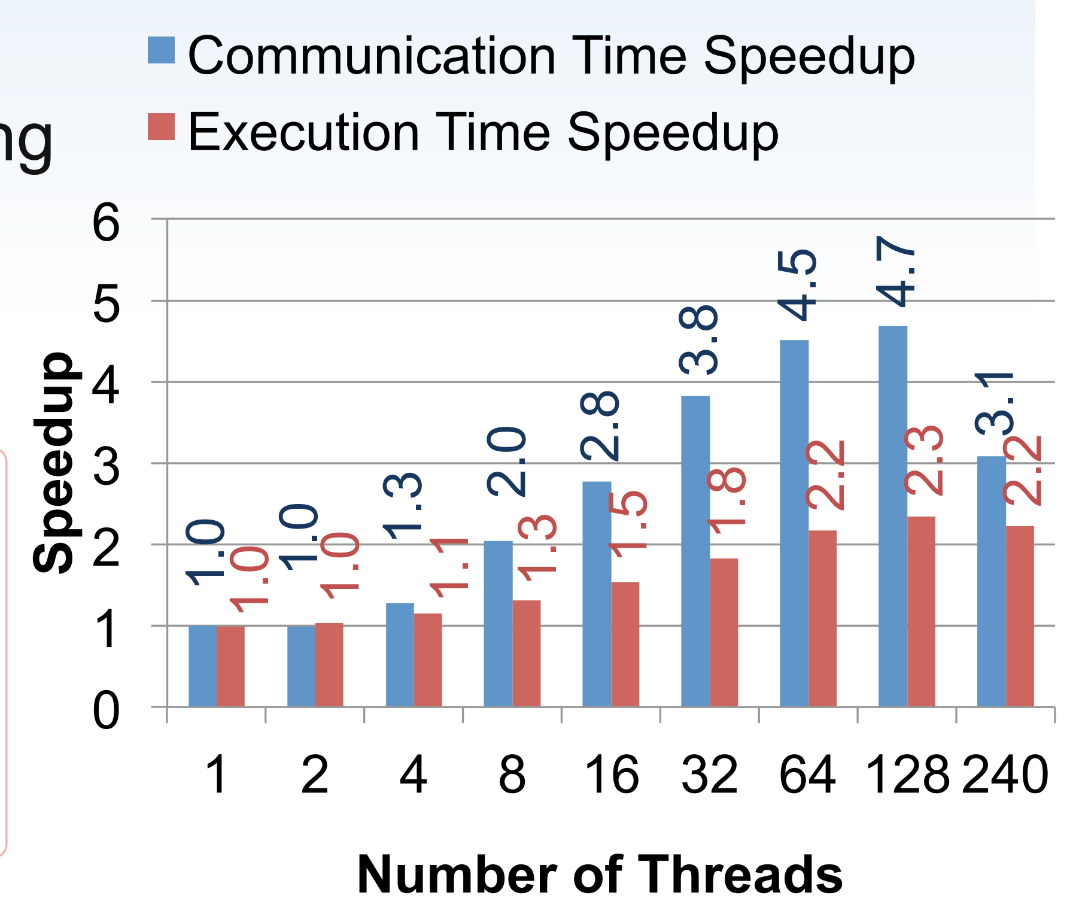
■ OpenMP Runtime Extension
- Expose **number of IDLE threads**

■ MPI Internal Parallelism
1. Derived datatype processing
2. SHM communication
3. InfiniBand network

Challenges

- **Algorithms tradeoff**
  - Number of IDLE threads is unknown.
- **Nested parallelism**
  - Creates new Pthreads and offloads scheduling to OS
  - Threads overrunning.

Hybrid NAS MG - class E using 64 MPI processes

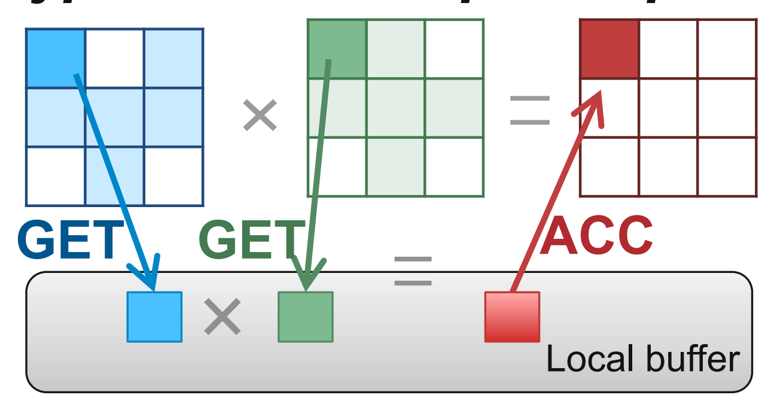


Part II. Process-based Asynchronous Progress Model for MPI RMA

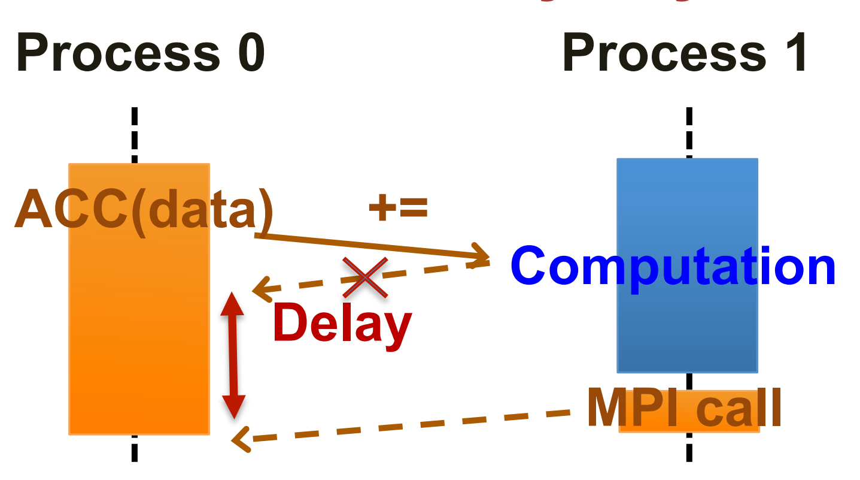
Problem in MPI RMA-based PGAS

- Suitable for applications with large memory requirements. (i.e., NWChem)

Typical Get-Compute-Update



- **MPI RMA is not truly asynchronous**



Traditional ASYNC Progress

1. Thread-based Approach
- Per-MPI process background thread

× Waste half cores or oversubscription
× Multithreading overhead in MPI

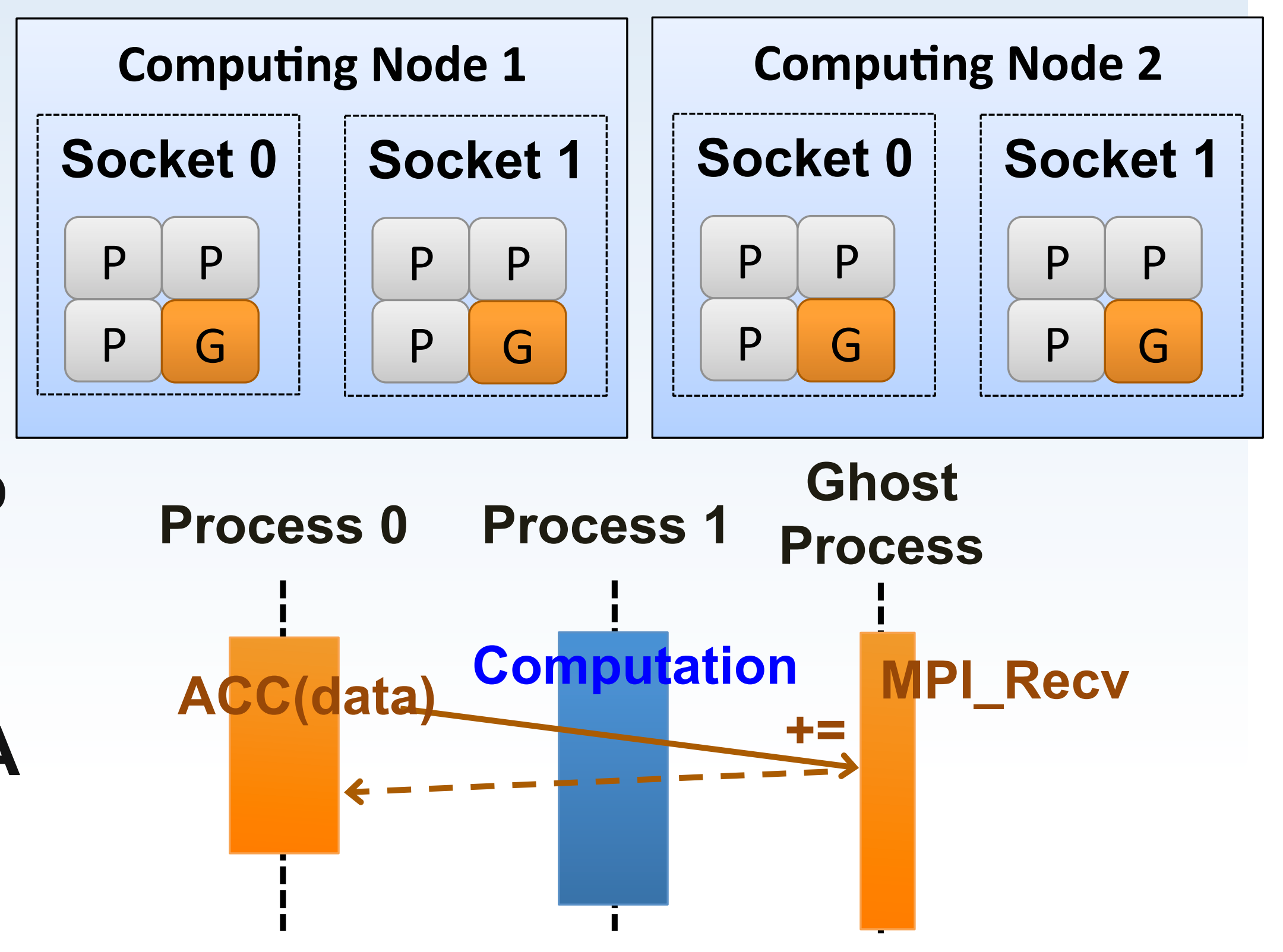
2. Interrupt-based Approach
- Per-operation hardware interrupts awaken kernel thread

× Overhead of frequent interrupts

Solution: Casper — Process-based Asynchronous Progress Model [2]

■ Core concept
- #cores is rapidly growing
- **Not all of the cores are always keeping busy**
- Dedicate **small & user-specified** number of cores to ghost processes
- **Ghost process** intercepts all RMA operations to the user processes
- Improve ASYNC progress for SW-handled operations **without affecting HW-handled RMA**

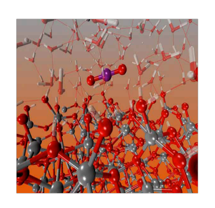
✓ **No multithreading or interrupt overhead**
✓ **Flexible core deployment**
✓ **Portable PMPI* redirection**



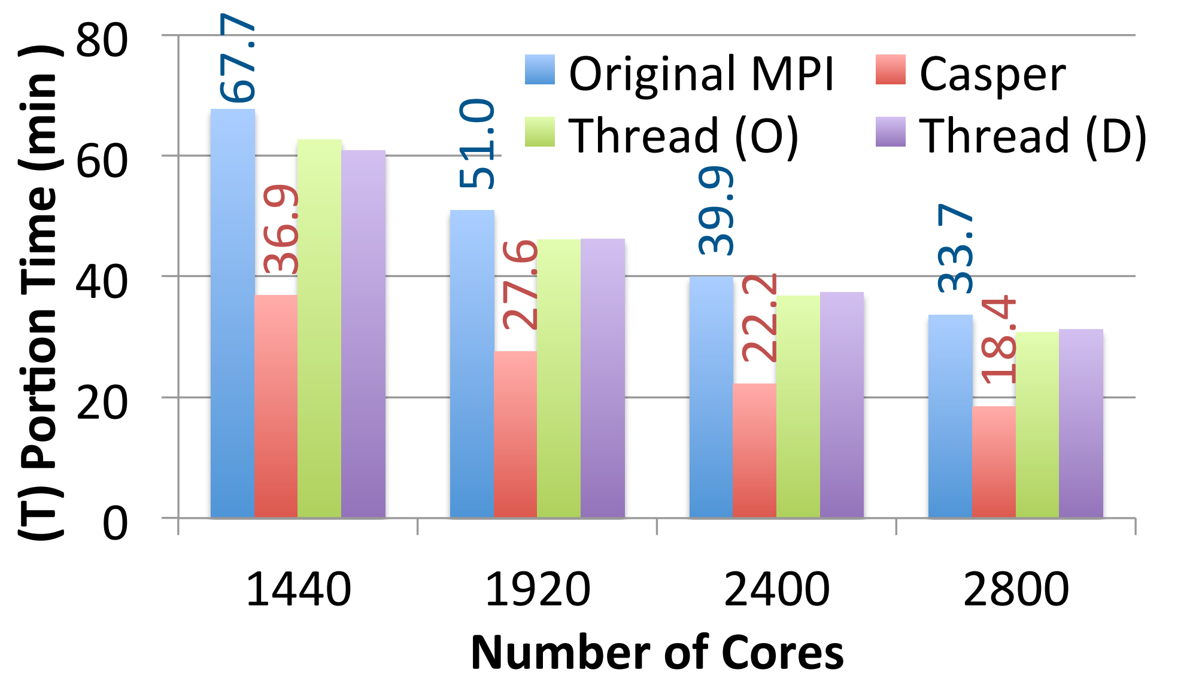
Correctness and Performance challenges

- **Ensuring correctness**
  1. Lock permission for shared ghost processes
  2. Managing multiple ghost processes
  3. Self lock consistency
  4. Multiple simultaneous epochs
- **Ensuring performance**
  1. Memory locality

NWChem Quantum Chemistry Suite



NERSC Edison Cray XC30
Input tce_c20_triplet
Compute-intensive CCSD(T) simulation



Reference
[1] M.Si, A.J.Pena, P.Balaji, M.Takagi, and Y.Ishikawa, “MT-MPI: Multithreaded MPI for Many-Core Environments,” in Proceedings of the 28th ACM ICS, June 2014.
[2] M.Si, A.J.Pena, J.Hammond, P.Balaji ,M.Takagi, and Y.Ishikawa, “Casper: An Asynchronous Progress Model for MPI RMA on Many- Core Architectures,” in IPDPS 2015.

Argonne National Laboratory is a U.S. Department of Energy laboratory managed by U Chicago Argonne, LLC.

The experimental resources for this research were provided by the Texas Advanced Computing Center (TACC) on the Stampede supercomputer and by the National Energy Research Scientific Computing Center (NERSC) on the Edison Cray XC30 supercomputer. This material was based upon work supported by the U.S. Dept. of Energy, Office of Science, Advanced Scientific Computing Research (SC-21), under contract DE-AC02-06CH11357